Deep Learning for Natural Language Processing

Mike Lewis

Facebook AI Research

4/20/20

Deep Learning for NLP

- Amazing progress in recent years:
  - Humans prefer machine translation to human translators for some languages
  - Super human performance on many question answering datasets
  - Language models generate fluent paragraphs

- *Minimal specialist techniques needed per task*

Language Models

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Radford et al. (2019)

Pérez and the others then ventured further into the valley. “By the time we reached the top of one peak, the water looked blue, with some crystals on top,” said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, “We can see, for example, that they have a common ‘language,’ something like a dialect or dialectic.”

Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America.

Language Models

- Language models assign a probability to a text

  $p(x_0, \ldots, x_n)$

- There are *many* possible sentences
  - We can't just train a classifier

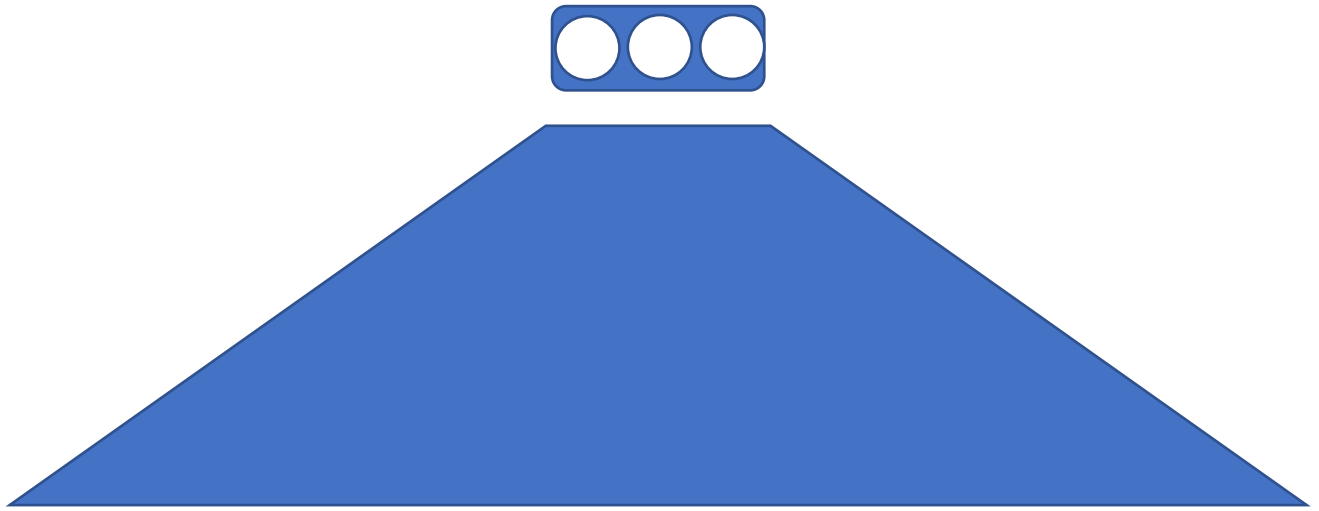
Language Models

- Factorize distribution using chain rule:
  - $p(x_0, \ldots, x_n) = p(x_0) * p(x_1 \mid x_0) * p(x_2 \mid x_1, x_0)$
- Now we have lots of classification problems like:
  - p(*unicorn* | *The scientist named the population, after their distinctive horn, Ovid's*)

Neural Language Models

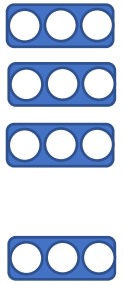
p(*unicorn* | *The scientist named the population, after their distinctive horn, Ovid's*)

Context Encoder *f*



The scientist named the population, after their distinctive horn, Ovid's

Word Embedding Matrix *E*



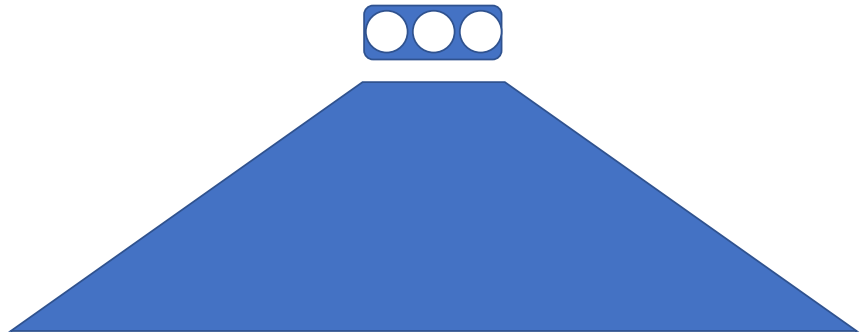
the
dog
three
…
unicorn
…

Neural Language Models

p(*unicorn* | *The scientist named the population, after their distinctive horn, Ovid's*)

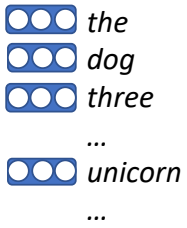
$$p(x_n \mid x_{0..n-1}) = \text{softmax}(E\ f(x_{0..n-1}))$$

Context Encoder *f*

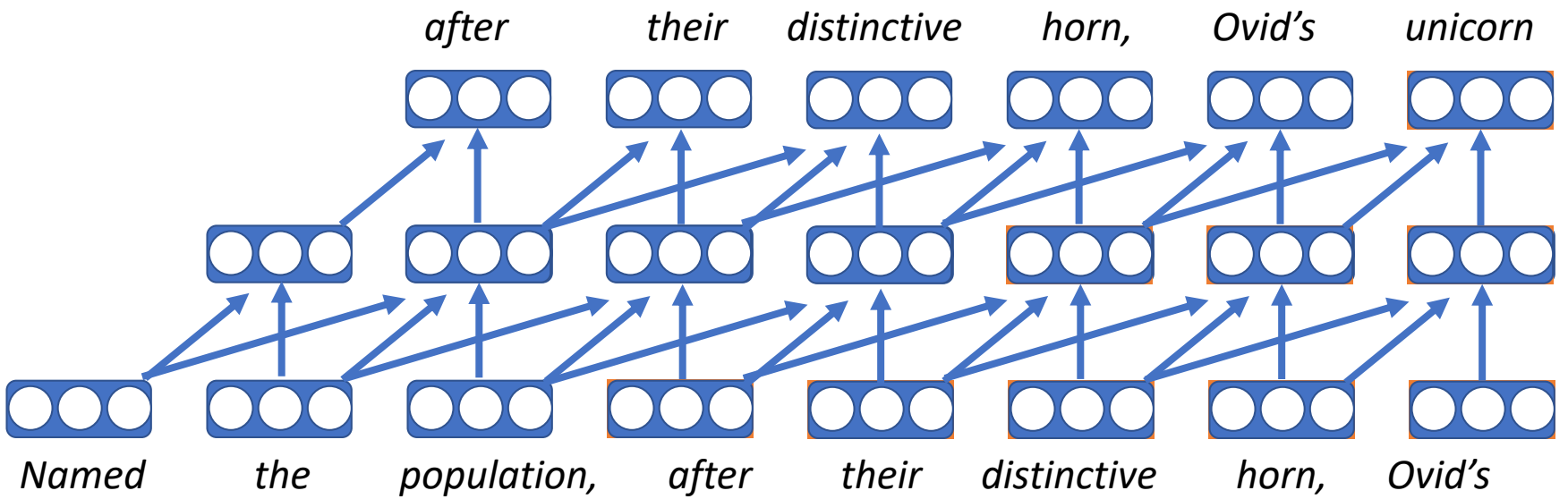


The scientist named the population, after their distinctive horn, Ovid's)

Word Embedding Matrix *E*



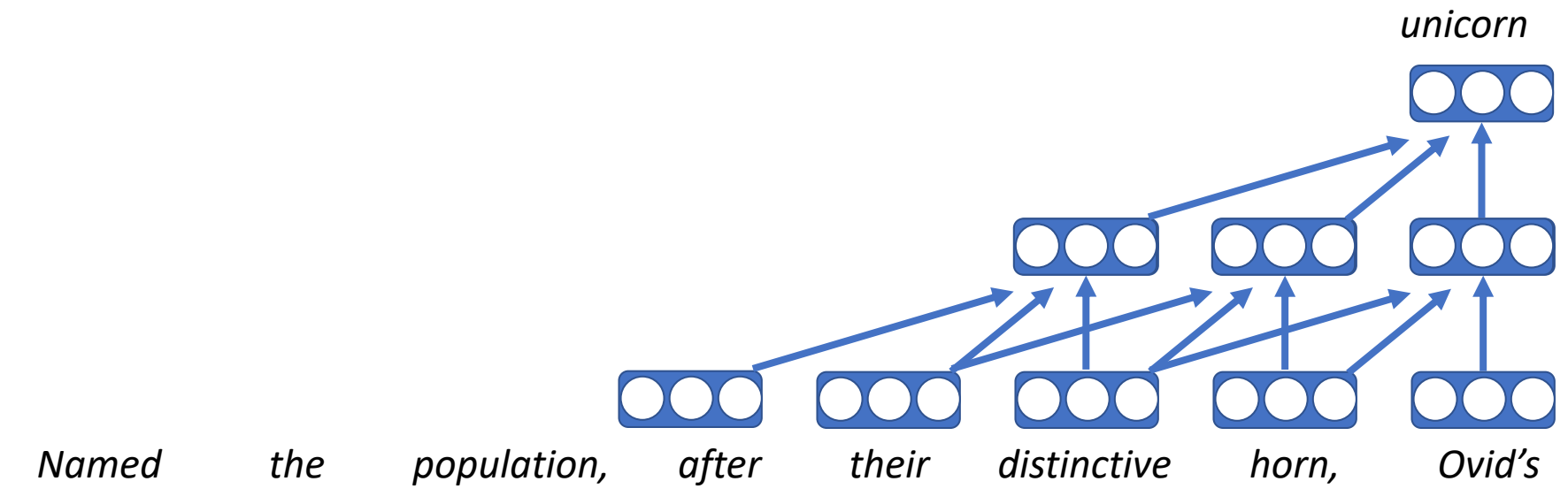
Convolutional Language Models



- Embed each word as a vector
- Apply same feed forward network at each timestep
- Fixed length history means it can only condition on bounded context

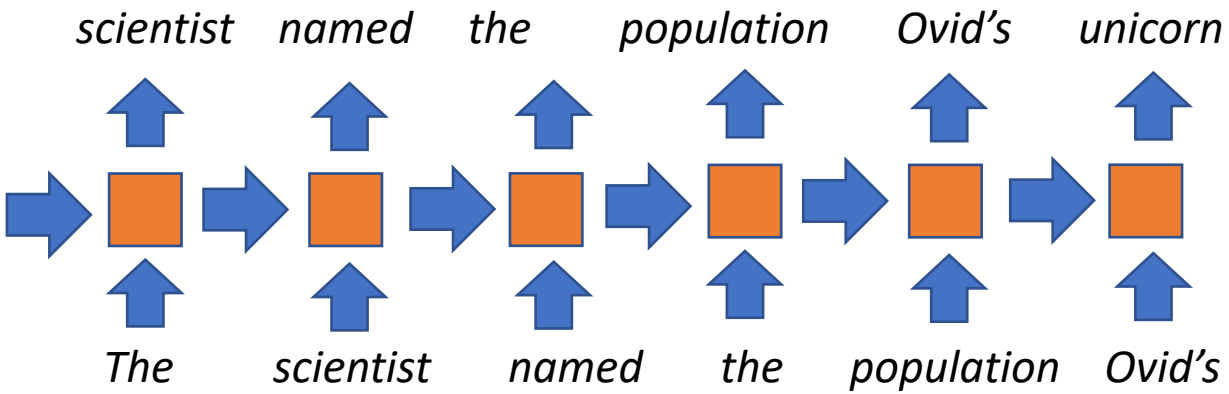
Bengio et al. (2003), Dauphin et al. (2016)

Convolutional Language Models



- Embed each word as a vector
- Apply same feed forward network at each timestep
- Fixed length history means it can only condition on bounded context

Recurrent Language Models

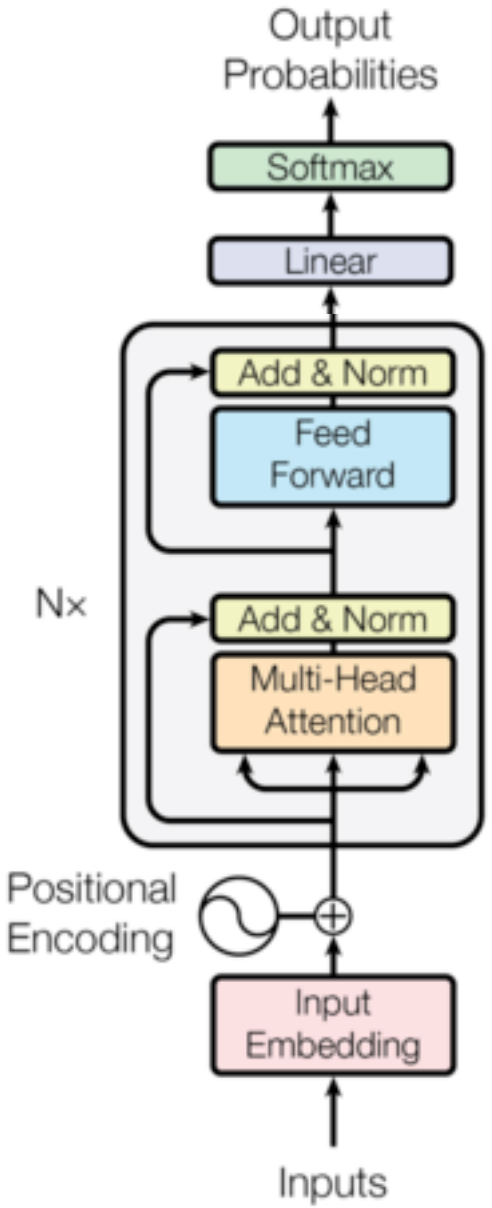


- Uses unbounded context!
- Context is compressed into fixed-size vector at each timestep
- Gradients tend to vanish with long contexts
- Not possible to parallelize over time-steps, so slow training

Mikolov et al. (2010)

Transformer Language Models

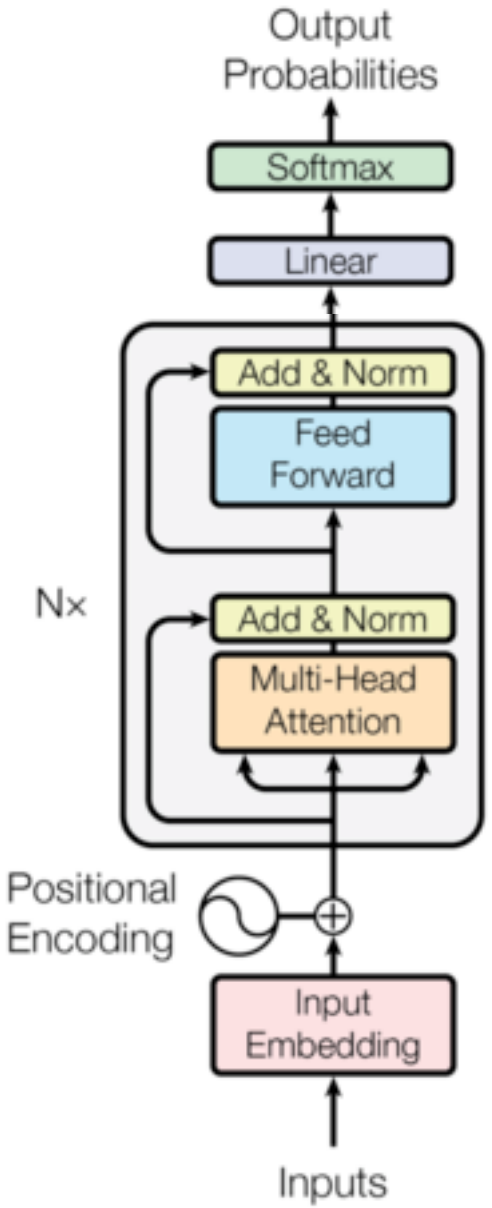
- *Attention is all you need!*

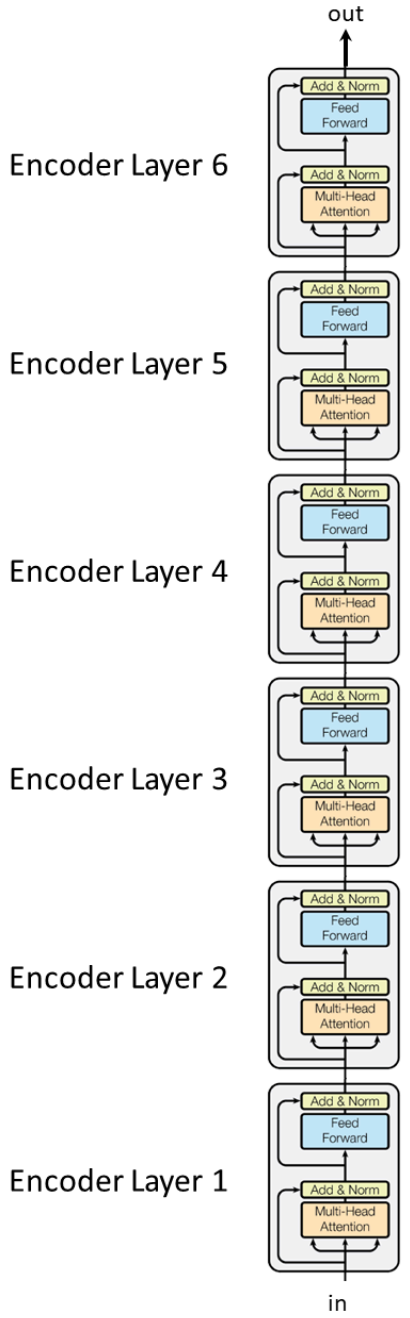


Vaswani et al. (2017)

Transformer Language Models

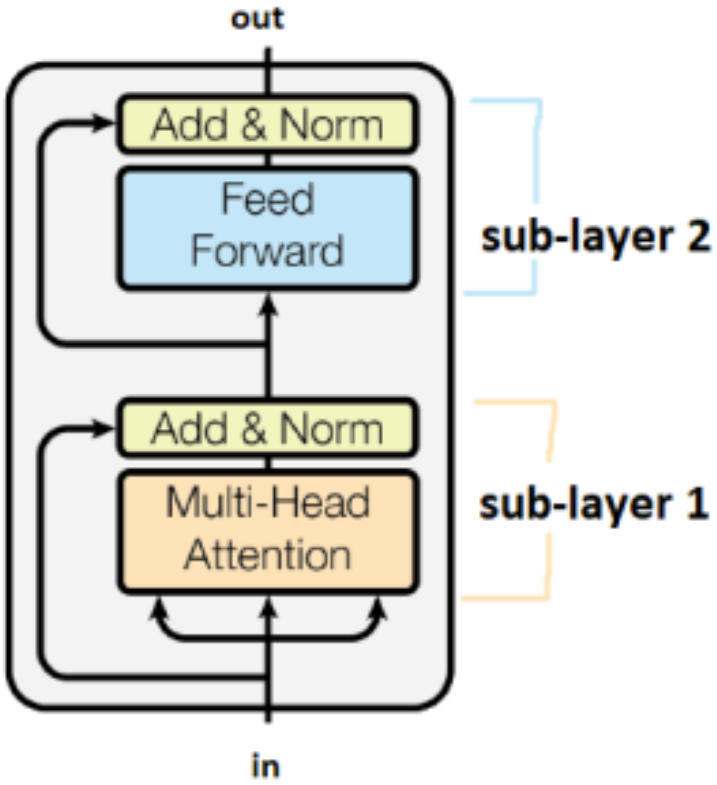
- *Attention is all you need*





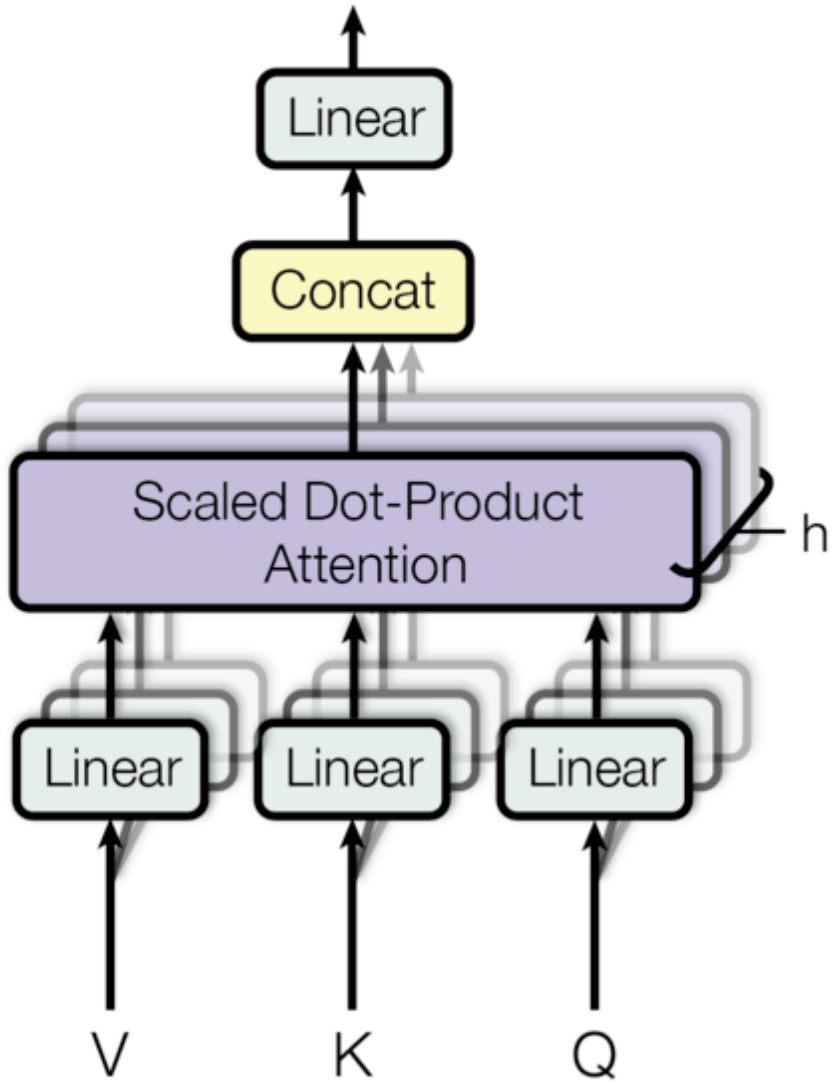
Transformer Language Models

- *Attention is all you need*



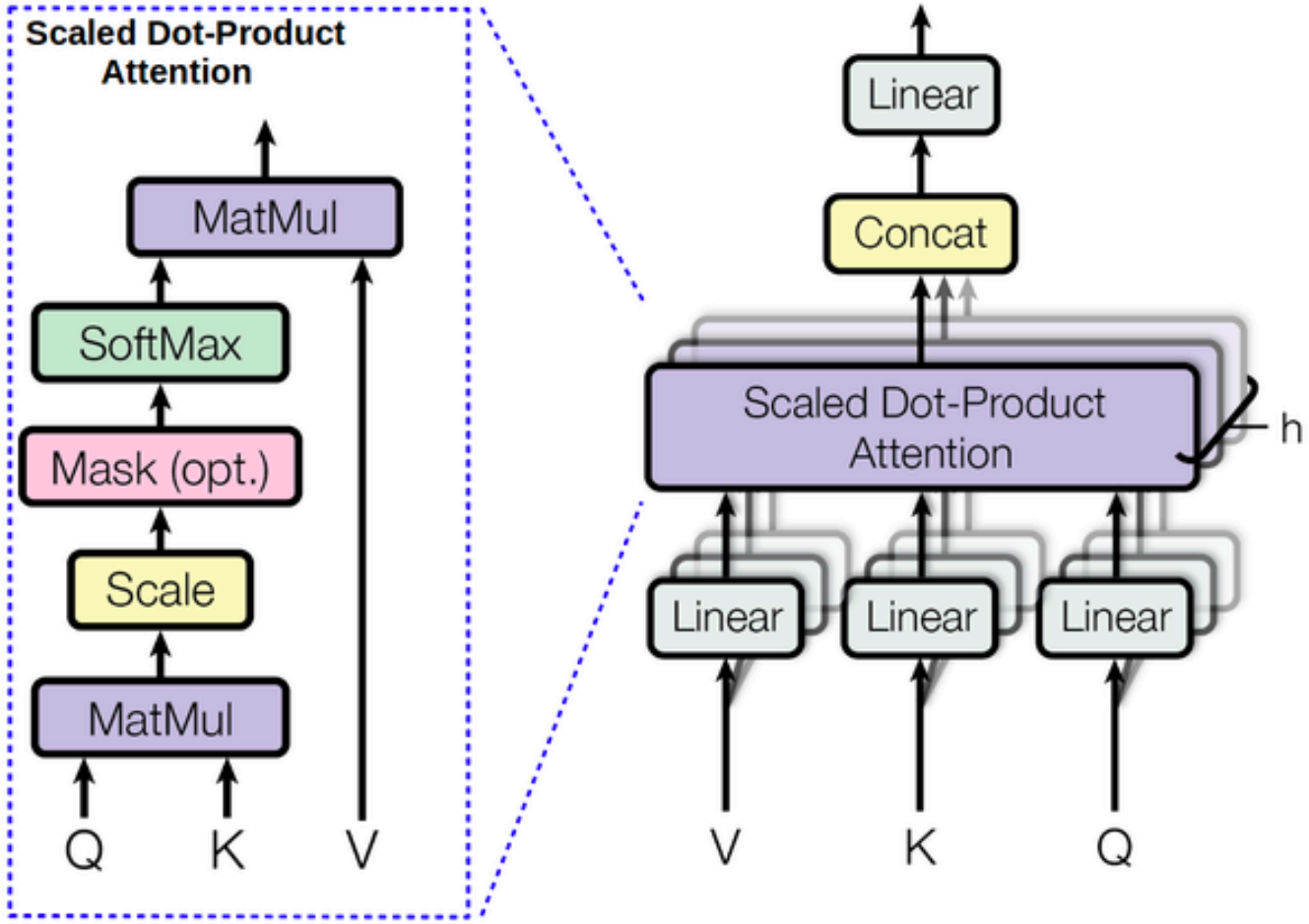
Transformer Language Models

- *Attention is all you need*



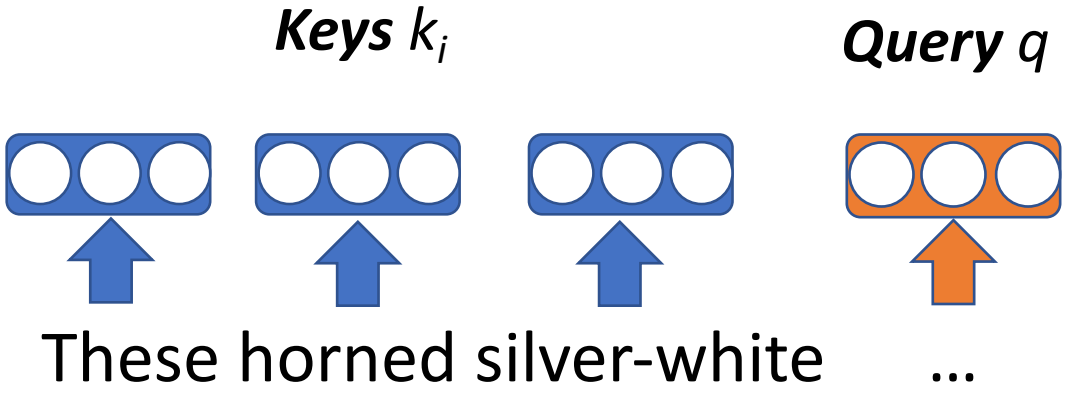
Transformer Language Models

- *Attention is all you need*



Transformer Language Models

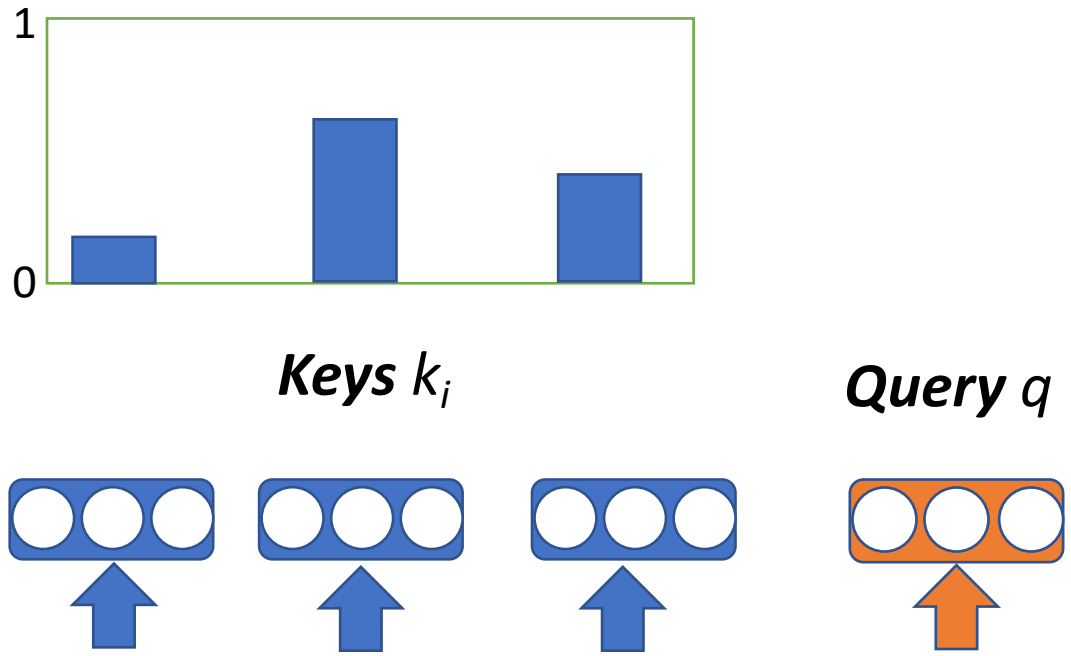
- Compute *key* and *value* for each previous word
- Compute query for current output
- Distribution over all previous words



Transformer Language Models

- Compute *key* and *value* for each previous word
- Compute query for current output
- Distribution over all previous words

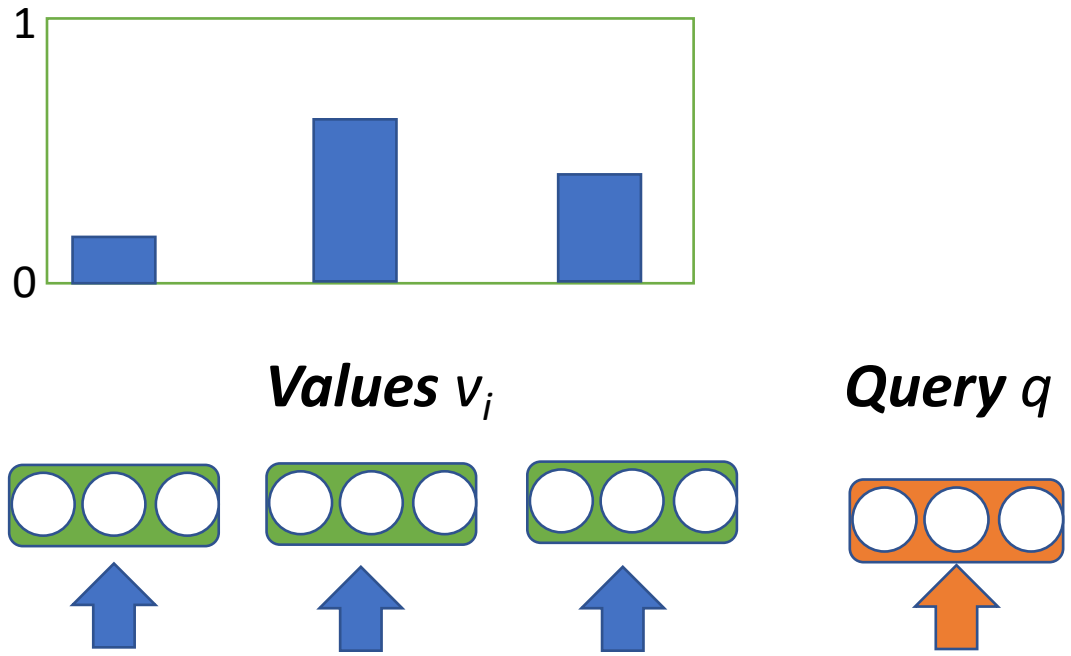
Attention distribution $p_i = softmax(q \cdot k_i)$



Transformer Language Models

- Compute *key* and *value* for each previous word
- Compute query for current output
- Distribution over all previous words

Attention distribution $p_i = softmax(q \cdot k_i)$

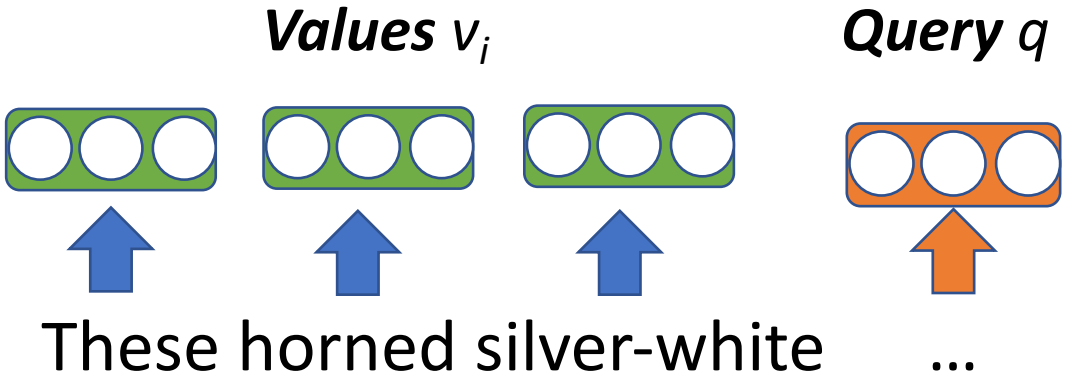


Transformer Language Models

- Compute *key* and *value* for each previous word
- Compute query for current output
- Distribution over all previous words

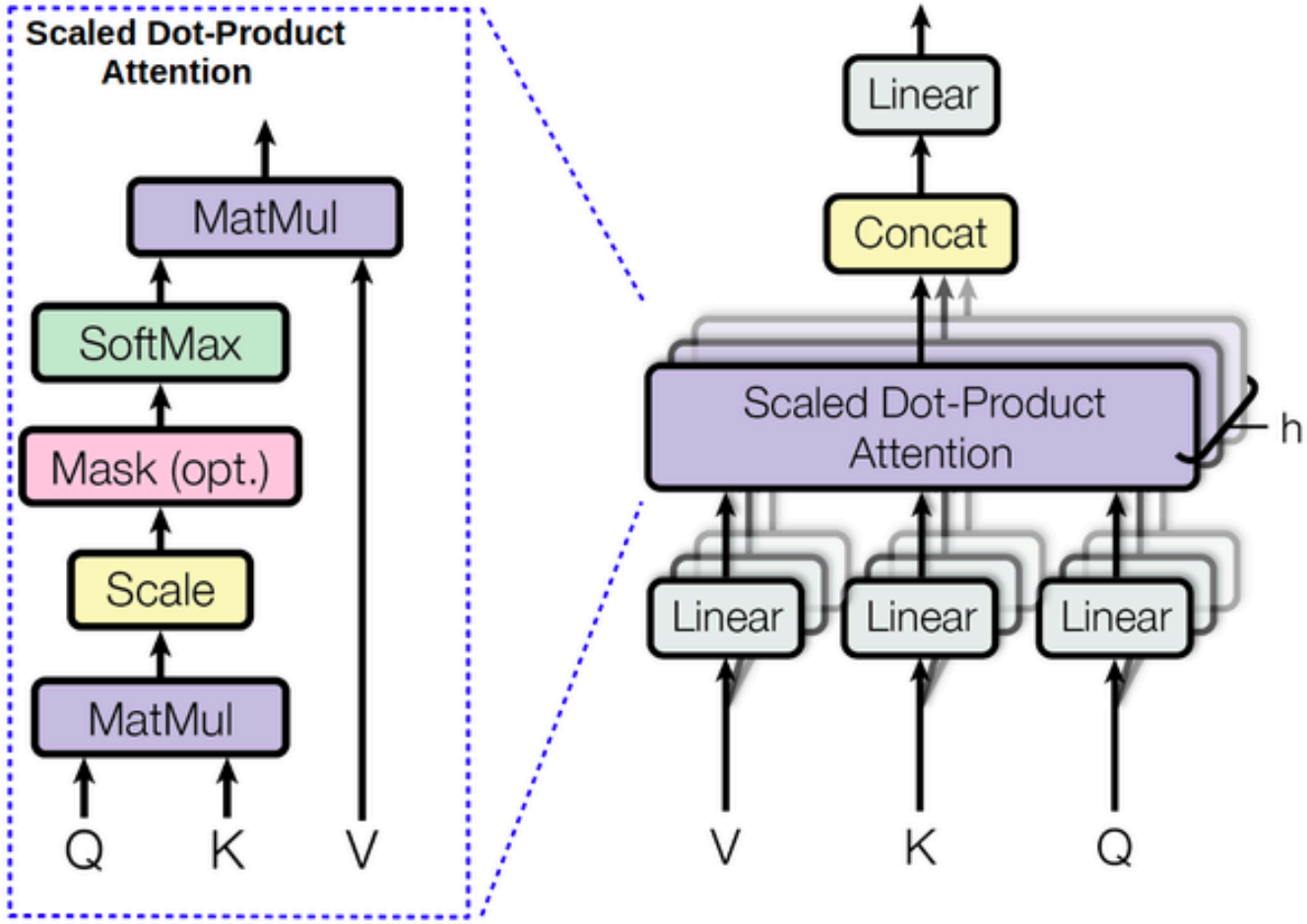
Attention distribution $p_i = softmax(q \cdot k_i)$

Attention Output $h_i = \sum_i p_i v_i$



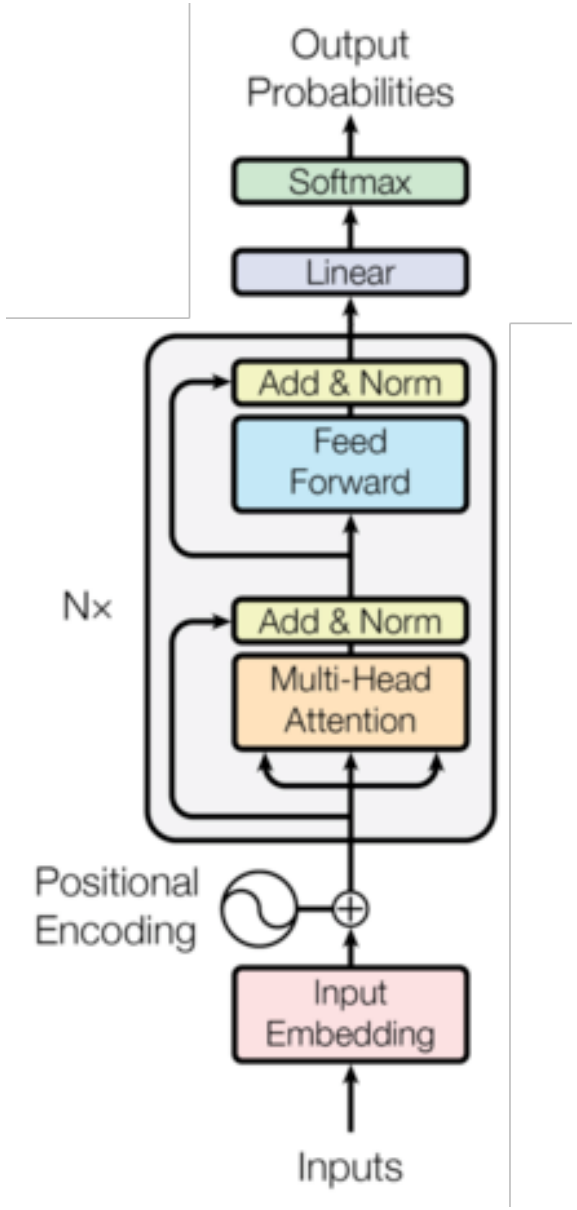
Transformer Language Models

- *Multi-head attention: do this all many times in parallel!*
- *Compute all time-steps in parallel*



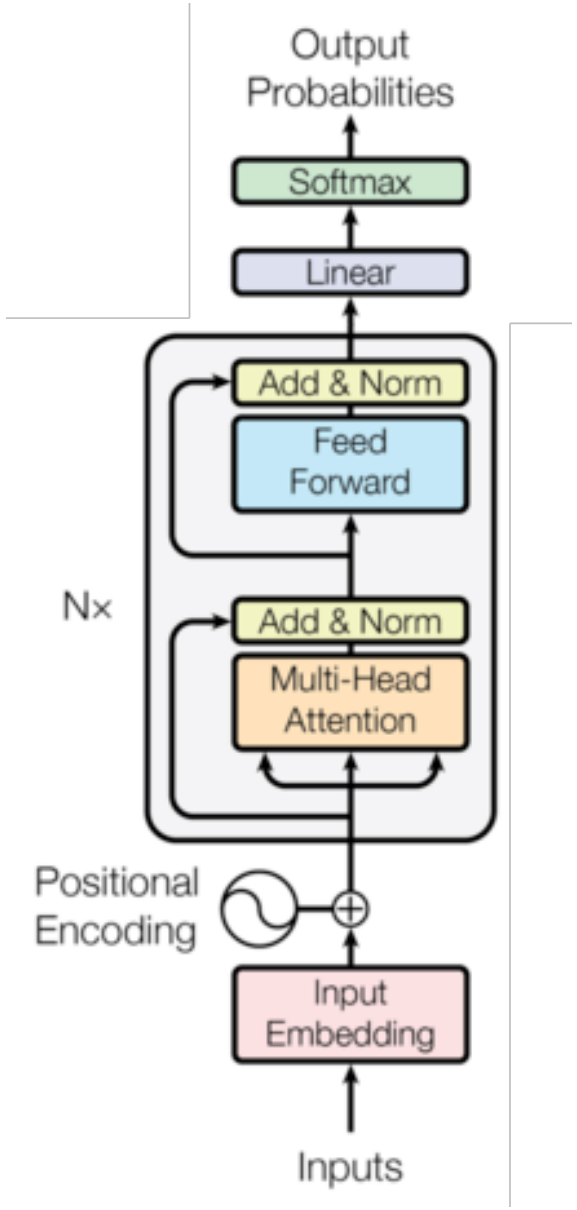
Transformer Language Models

- Self attention is order agnostic
  - Add *positional embedding* to input
  - Mask self attention so words cannot look into the future



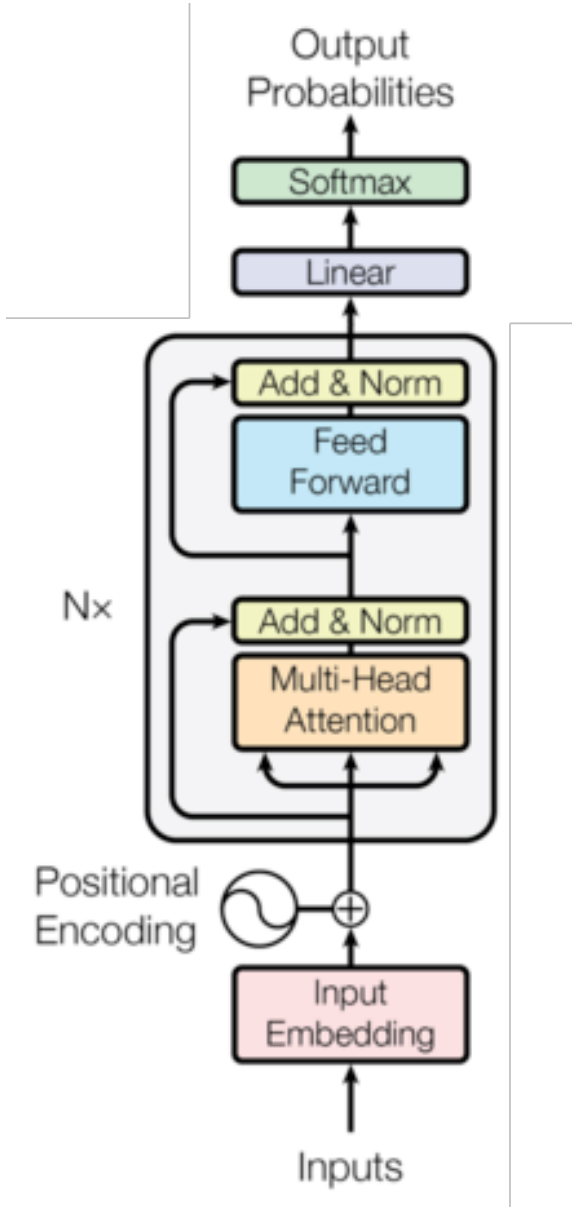
Transformer Language Models

- All words directly connected, mitigating vanishing gradients
- All time-steps computed in parallel
- Self attention is quadratic (all time-steps can attend to all others), limiting max sequence length



Transformer Language Models

- Lots of tricks:
  - Extensive use of layer normalization to stabilize training
  - Warmup + Inverse-square root training schedule
  - Careful initialization
  - Label smoothing



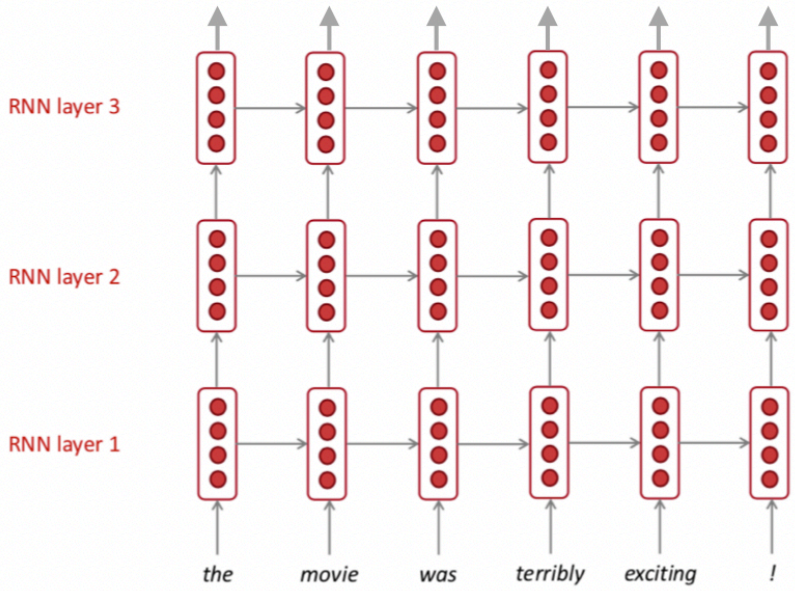
Transformer Language Models

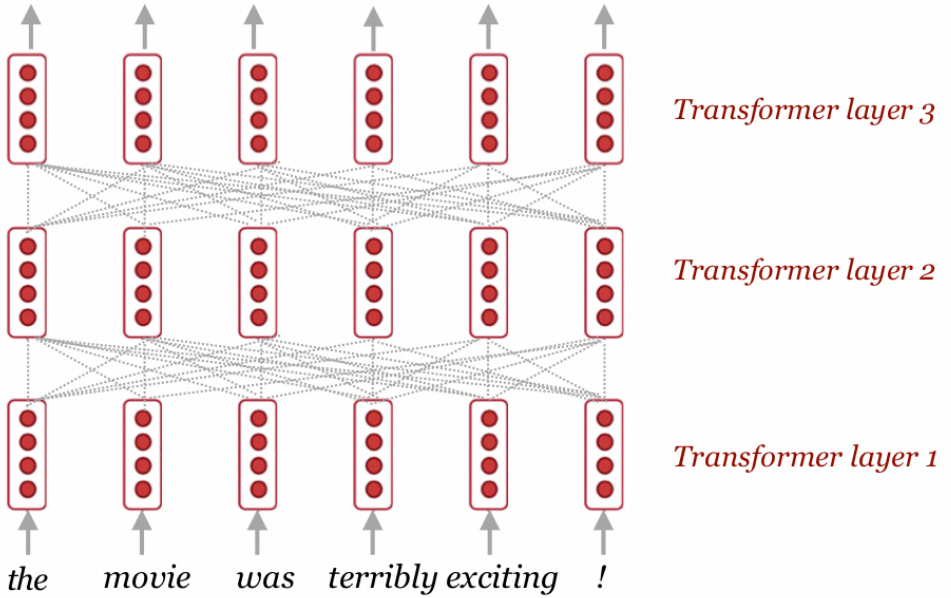
- Lots of tricks:
  - Extensive use of layer normalization to stabilize training
  - Warmup + Inverse-square root training schedule
  - Careful initialization
  - Label smoothing

Model	#Param	PPL
Grave et al. (2016b) - LSTM	-	48.7
Bai et al. (2018) - TCN	-	45.2
Dauphin et al. (2016) - GCNN-8	-	44.9
Grave et al. (2016b) - LSTM + Neural cache	-	40.8
Dauphin et al. (2016) - GCNN-14	-	37.2
Merity et al. (2018) - QRNN	151M	33.0
Rae et al. (2018) - Hebbian + Cache	-	29.9
Ours - Transformer-XL Standard	151M	**24.0**
Baevski and Auli (2018) - Adaptive Input$^{\diamond}$	247M	20.5
Ours - Transformer-XL Large	257M	**18.3**

Transformer Language Models

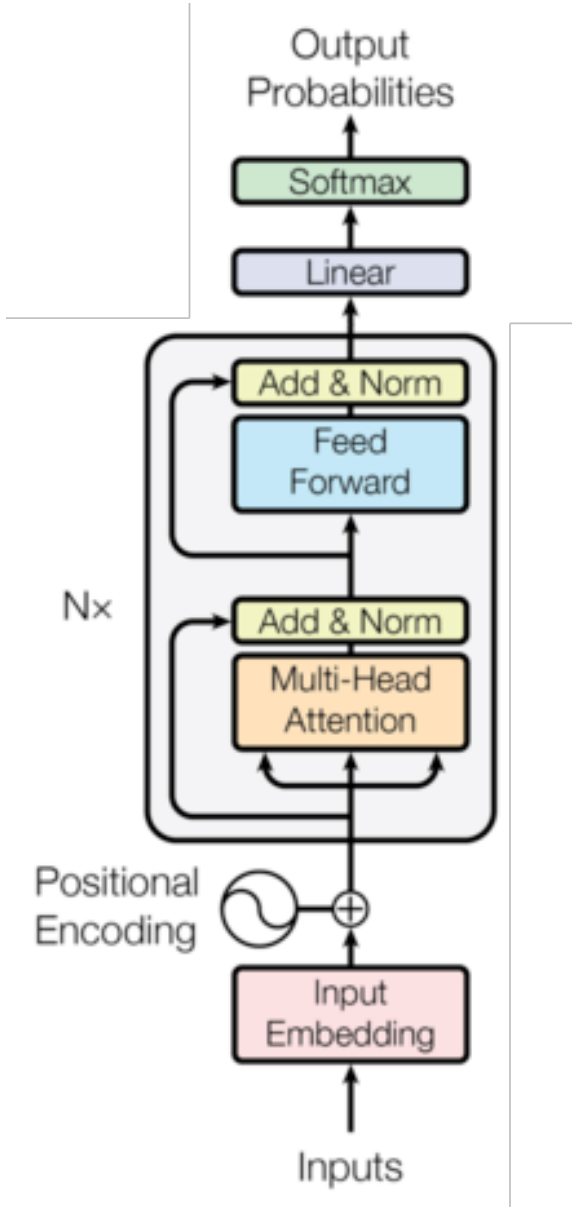
- Minimal inductive bias
- All words directly connected, mitigating vanishing gradients
- All time-steps computed in parallel
- Self attention is quadratic (all time-steps can attend to all others), limiting max sequence length





Transformer Language Models

- Transformers scale up well
  - Unlimited training data
  - GPT2 uses 2 billion parameters
  - Recent models use up to 17B parameters



Decoding Language Models

We can now train a probability distribution over text - now what?

- Exponentially many possible outputs, so can’t compute the max

Greedy Decoding

- Take most likely word at each timestep
  - No guarantee this gives most likely *sequence*

Exhaustive search?

- Find $\arg \underset{y_1,\ldots,y_T}{max}\ P(y_1,\ldots,y_T|x_1,\ldots,x_n)$
- Requires computing all possible sequences
  - $O(V^T)$ complexity!
  - Too expensive

(slide credit: Luke Zettlemoyer

A middle ground: Beam search

- **Key idea:** At every step, keep track of the k most probable partial translations (hypotheses)
- Score of each hypothesis = log probability

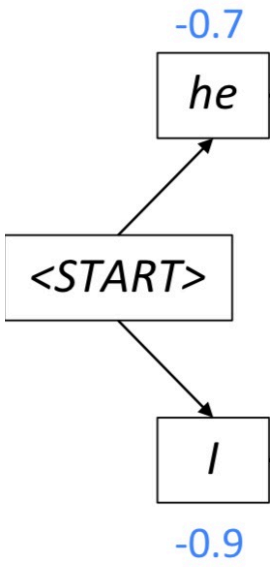
$$\sum_{t=1}^{j} \log P(y_t | y_1, \ldots, y_{t-1}, x_1, \ldots, x_n)$$

- Not guaranteed to be optimal
- More efficient than exhaustive search

(slide credit: Luke Zettlemoyer

Beam decoding

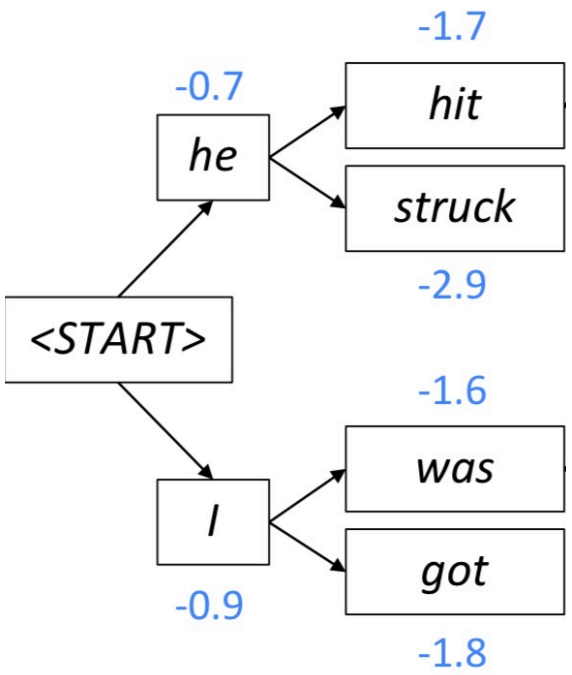
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

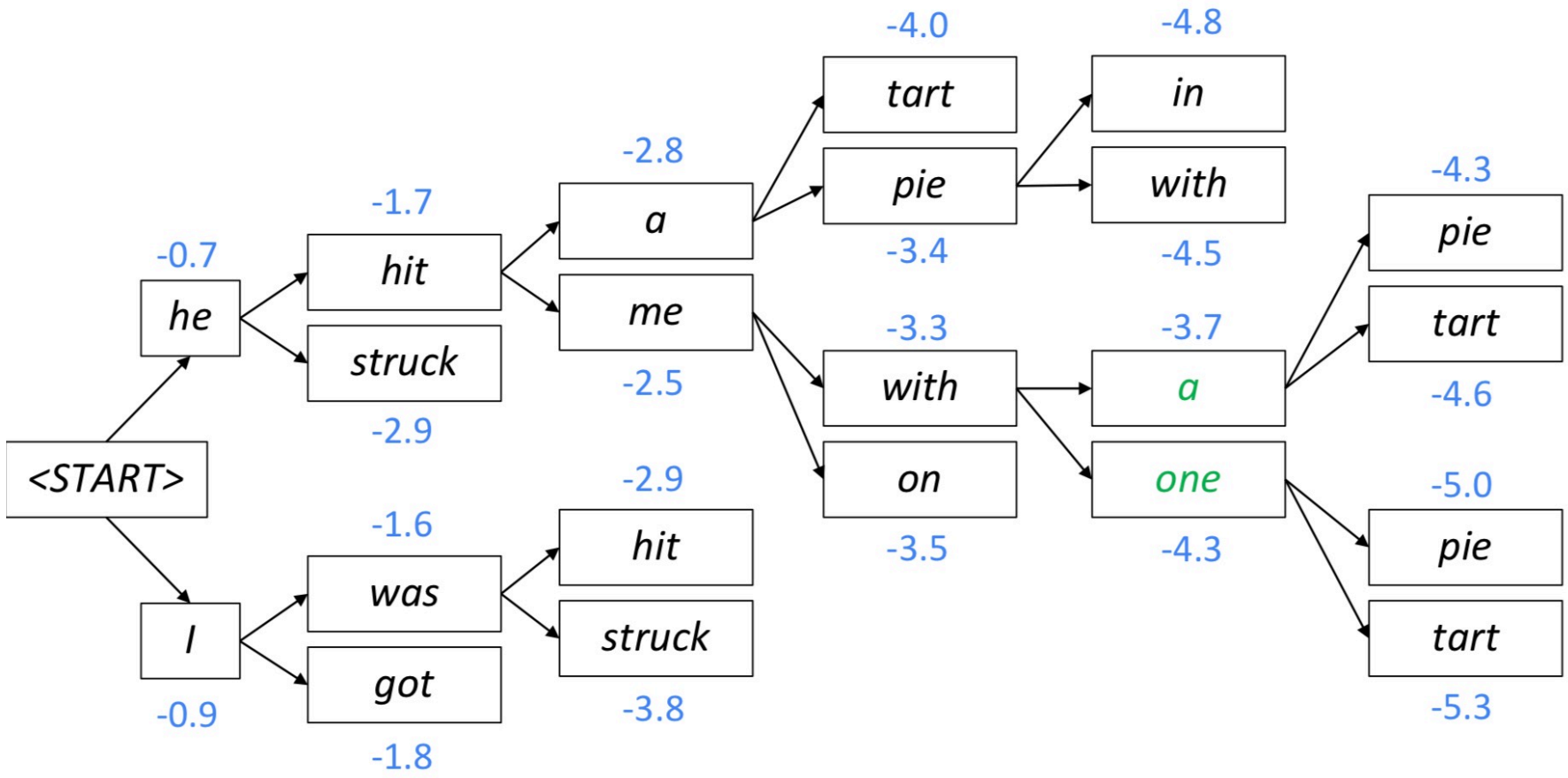
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

- Different hypotheses may produce $\langle e \rangle$ (end) token at different time steps
  - When a hypothesis produces $\langle e \rangle$, stop expanding it and place it aside
- Continue beam search until:
  - All k hypotheses produce $\langle e \rangle$ OR
  - Hit max decoding limit T
- Select top hypotheses using the *normalized* likelihood score

$$\frac{1}{T}\sum_{t=1}^{T}\log P(y_t|y_1,\ldots,y_{t-1},x_1,\ldots,x_n)$$

  - Otherwise shorter hypotheses have higher scores

(slide credit: Luke Zettlemoyer

Beam Search

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Example from Holtzman et al. 2020

Sampling

- Maybe we don’t want the most likely sequence after all
  - Instead, just sample from the model distribution

Sampling

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town, and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

Example from Holtzman et al. 2020

Sampling

- Maybe we don’t want the most likely sequence after all
  - Instead, just sample from the model distribution

- Once we sample one bad word, the model is in a state it never saw during training
  - Increases the chance of the next sample being bad…

Top-k Sampling

- Truncate distribution to top-k, then renormalize and sample
- Allows some diversity, while reducing risk of bad samples

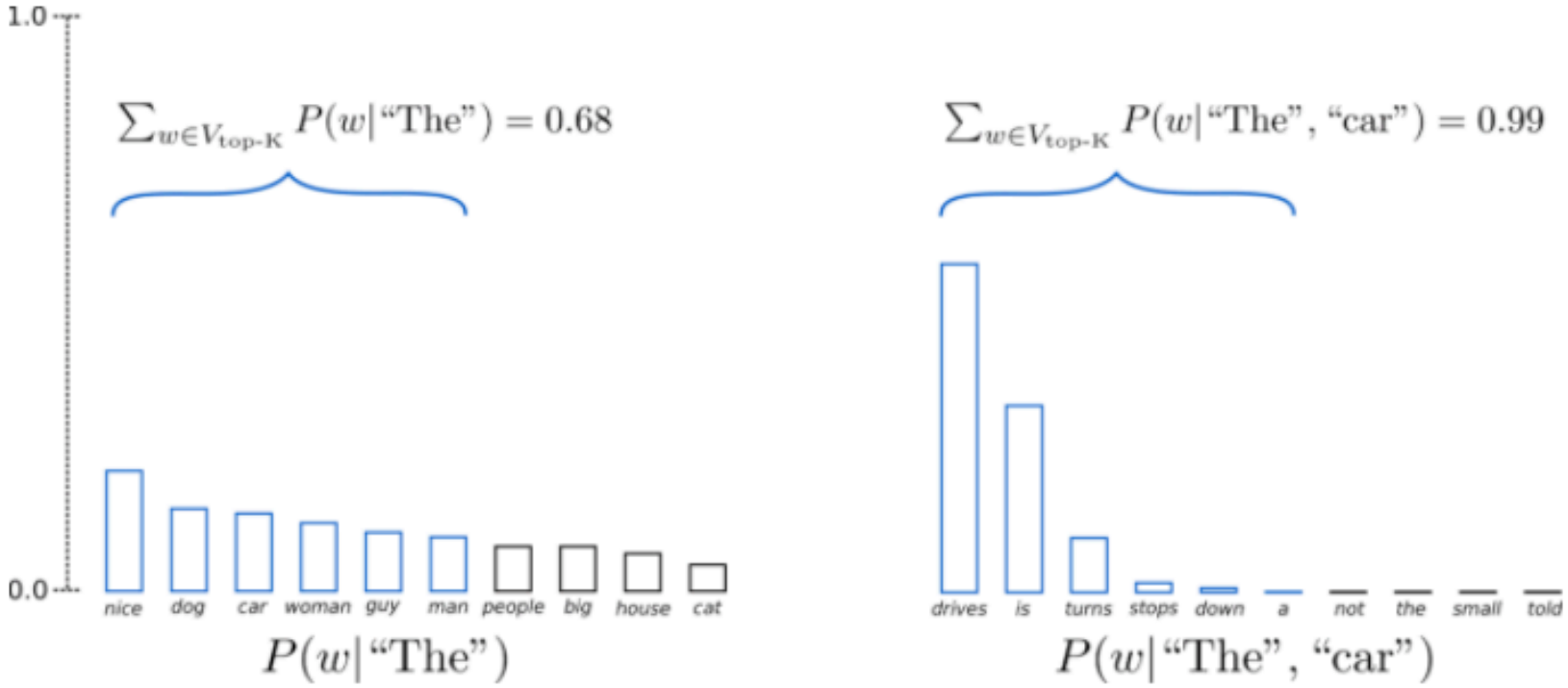


Figure from https://huggingface.co/blog/how-to-generate

Finally…

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Evaluating Text Generation

- Evaluating language models is easy – just measure likelihood of held out data

- Evaluating generated text is hard
  - Trade-off between quality and diversity
  - Word overlap metrics with a reference (BLEU, ROUGE etc.) are obviously wrong, but surprisingly useful
  - New automated metrics have been proposed

Sequence-to-Sequence Models

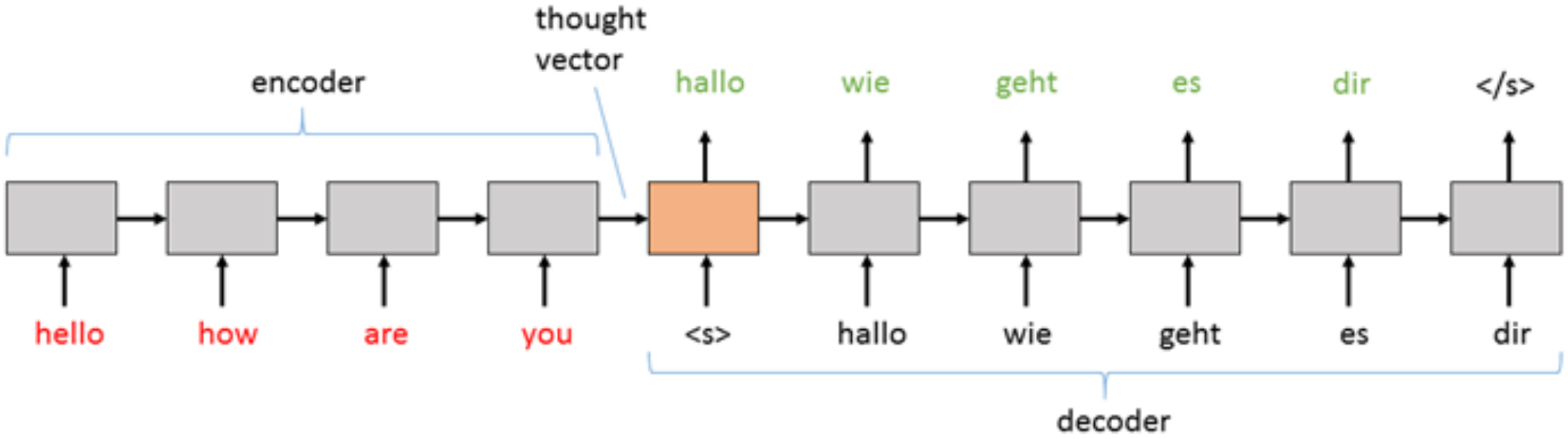
Conditional Language Models

- Generating random samples of English isn’t that useful

- Generate text given input:
  - Given a French sentence, generate the English translation
  - Given a document, generate a summary
  - Given a dialogue, generate the next response
  - Given a question, generate the answer

Sequence to Sequence Models

- Encoder module reads input
- Decoder module writes output word by word



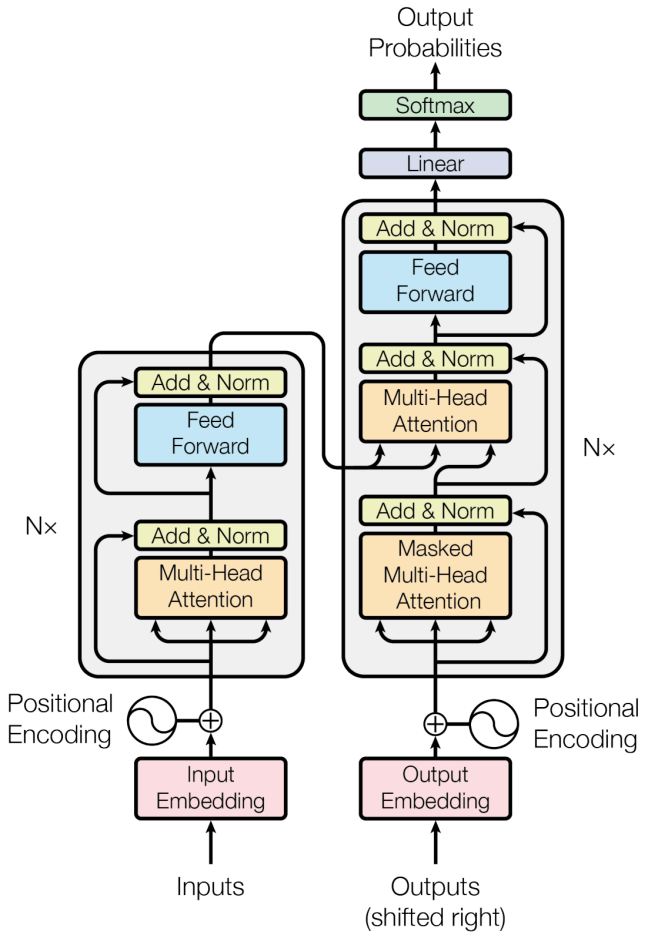
Sutskever et al. (2014)

Sequence-to-Sequence Transformers

Encoder Stack

Each input word attends to:

- All other input words



Decoder Stack

Each input word attends to:

- all input words
- All *previous* output words

Transformers

- Big = 6 layers, 1000 dim for each token, 16 heads, base = 6 layers + other params halved

Model	BLEU	
	EN-DE	EN-FR
ByteNet [18]	23.75	
Deep-Att + PosUnk [39]		39.2
GNMT + RL [38]	24.6	39.92
ConvS2S [9]	25.16	40.46
MoE [32]	26.03	40.56
Deep-Att + PosUnk Ensemble [39]		40.4
GNMT + RL Ensemble [38]	26.30	41.16
ConvS2S Ensemble [9]	26.36	**41.29**
Transformer (base model)	27.3	38.1
Transformer (big)	**28.4**	**41.8**

Vaswani et al. (2017)

Backtranslation

- Models are trained on parallel text of input and outputs
  - For example, European parliament proceedings
- Transformers do better with more data
  - Can we use unlimited amounts of *monolingual* text?

Backtranslation

- Example: we want to train a German to English translation model
- First, train an English to German *reverse* translation model using bitext
- Then, translate billions of words of monolingual English to German
- Finally, train German to English model using ‘back-translated’ data

Iterated Backtranslation

Description	My → En	En → My
Baseline (ensemble)	25.1	35.9
+ reranking	27.7	36.9
+ iter. 1 of ST + BT	35.5	40.1
+ iter. 2 of ST + BT	36.9	40.4
+ iter. 3 of ST + BT	37.9	40.6

Table 4: BLEU scores of systems trained using additional monolingual data.

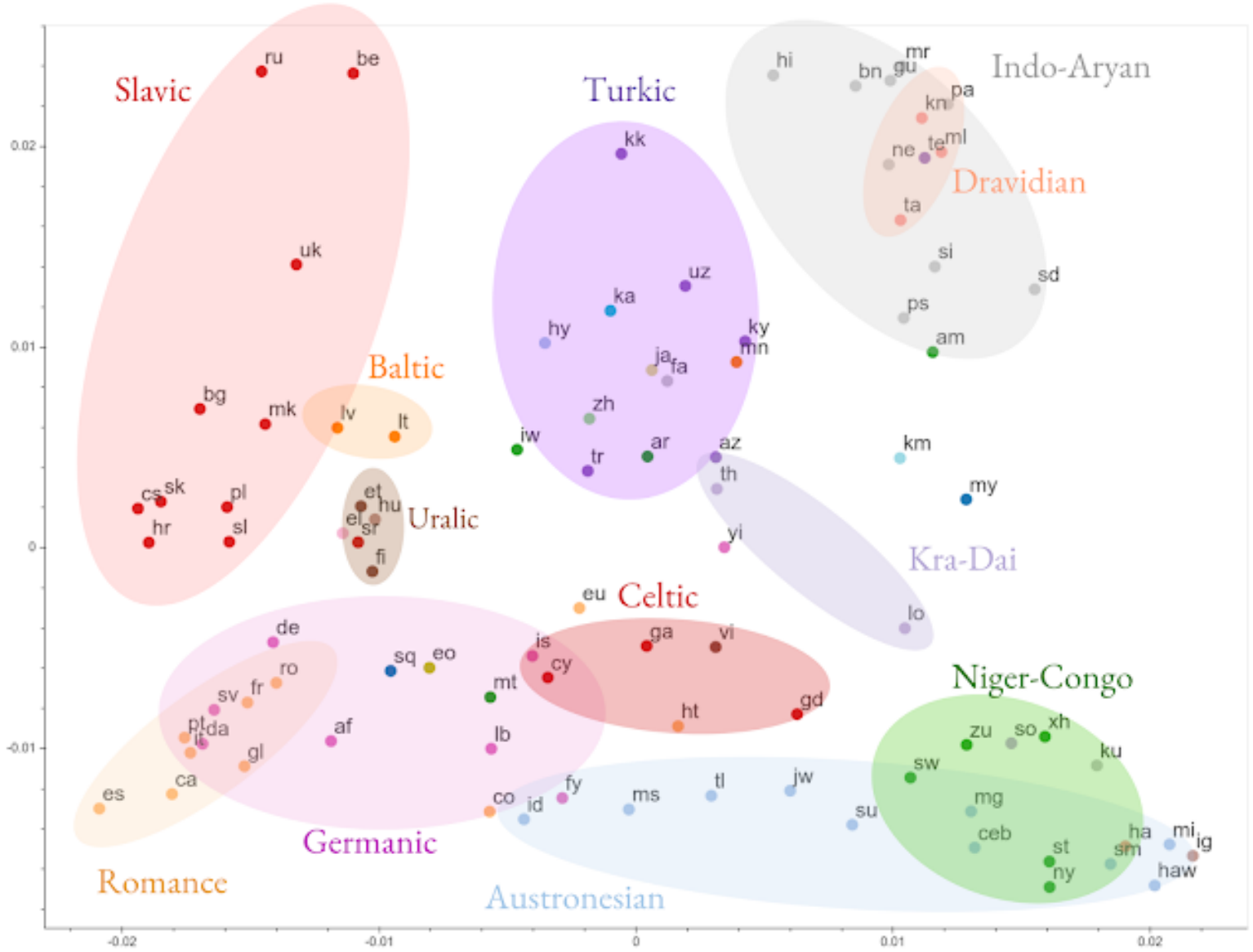
Backtranslation

name	training data	instances	BLEU tst2011	tst2012	tst2013	tst2014
baseline (Gülçehre et al., 2015)			18.4	18.8	19.9	18.7
deep fusion (Gülçehre et al., 2015)			20.2	20.2	21.3	**20.6**
baseline	parallel	7.2m	18.6	18.2	18.4	18.3
$\text{parallel}_{\text{synth}}$	parallel/$\text{parallel}_{\text{synth}}$	6m/6m	19.9	20.4	20.1	20.0
$\text{Gigaword}_{\text{mono}}$	parallel/$\text{Gigaword}_{\text{mono}}$	7.6m/7.6m	18.8	19.6	19.4	18.2
$\text{Gigaword}_{\text{synth}}$	parallel/$\text{Gigaword}_{\text{synth}}$	8.4m/8.4m	**21.2**	**21.1**	**21.8**	20.4

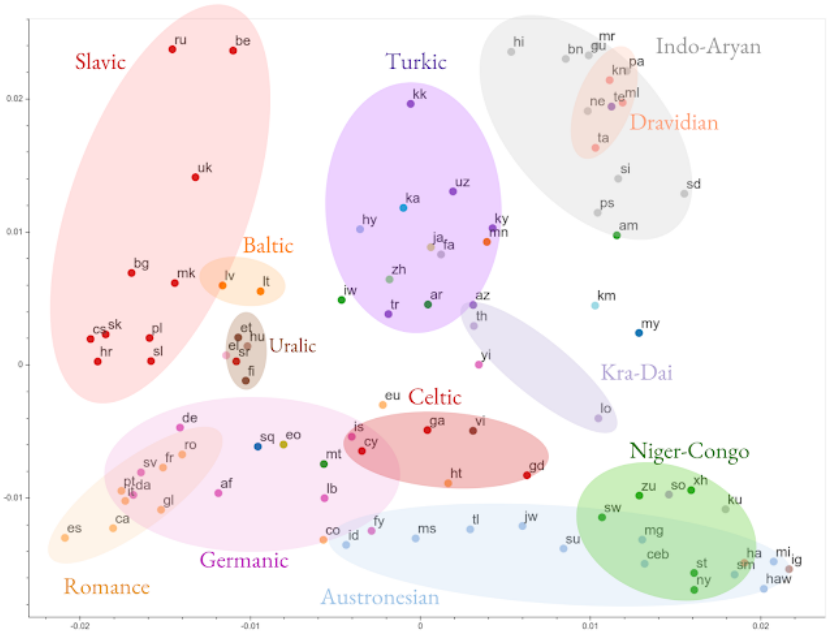
- Gigaword: large monolingual English corpus
- $\text{parallel}_{\text{synth}}$: backtranslate training data; makes additional noisy source sentences which could be useful

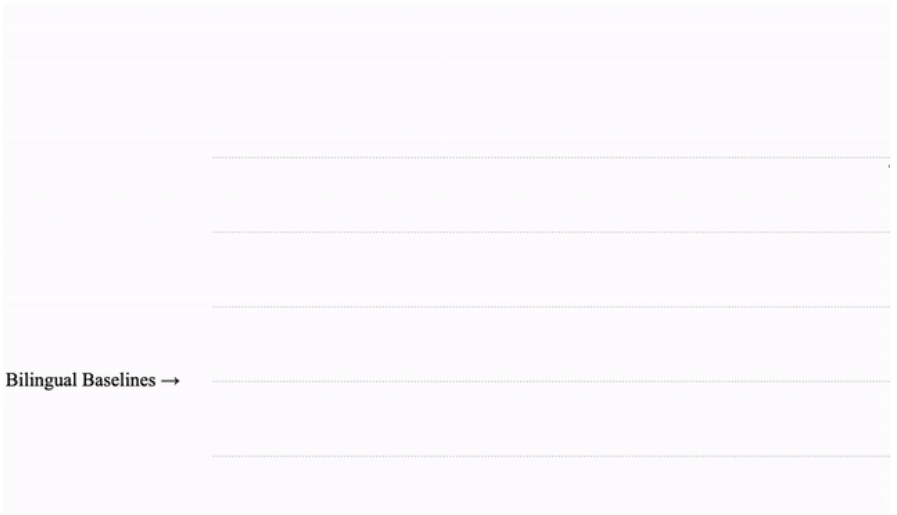
Sennrich et al. (2015)

Massively multilingual MT



Massively multilingual MT





- Train a *single* neural network on 103 languages paired with English
- Massive improvements on low-resource languages

(Arivazhagan et al., 2019)

Self-Supervised Learning

Self Supervised Learning

How much can we learn about language by just reading unlabeled text?

word2vec

- Learn vector space representations for words from unlabeled text
- Intuition: related words occur close to each other
- Use neighbouring words to fill in the blanks

These horned silver-haired unicorns were previously unknown

Mikolov et al. (2034)

word2vec

- Learn vector space representations for words from unlabeled text
- Intuition: related words occur close to each other
- Use neighbouring words to fill in the blanks

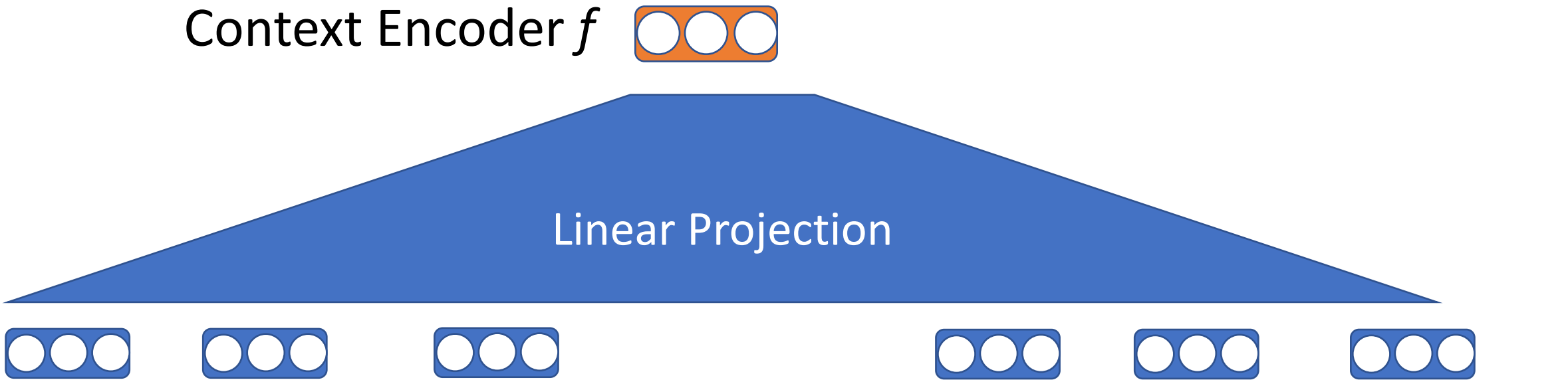
These horned silver-haired unicorns were previously unknown

word2vec

- Learn vector space representations for words from unlabeled text
- Intuition: related words occur close to each other
- Use neighbouring words to fill in the blanks

These horned silver-haired ??? were previously unknown

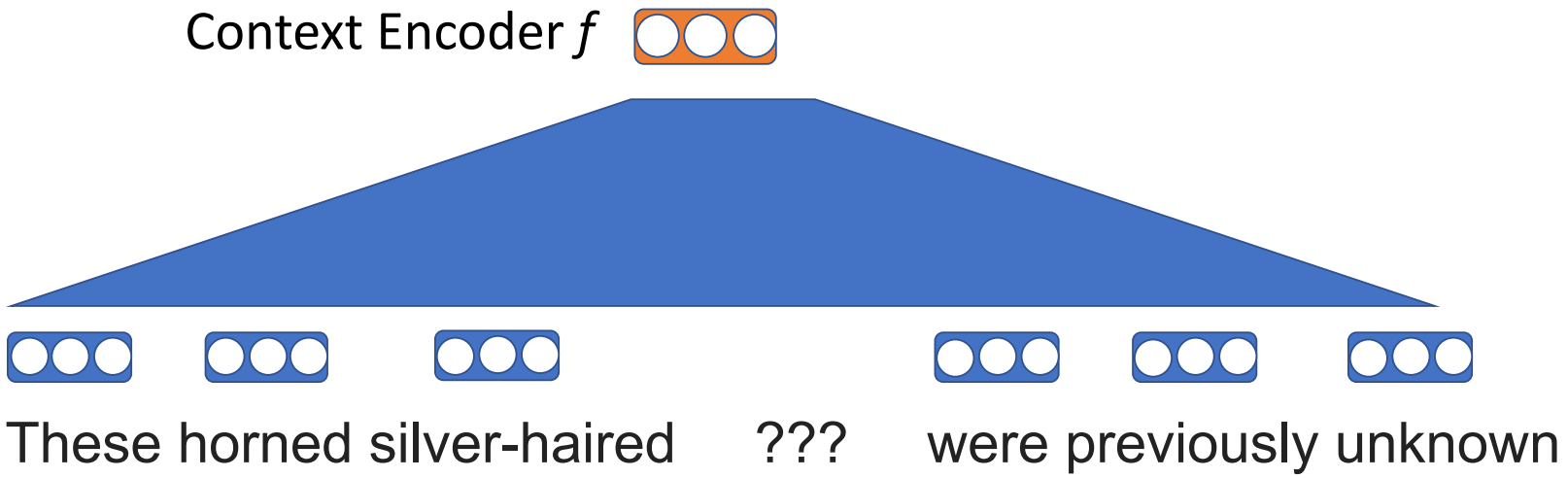
word2vec


Context Encoder f
Linear Projection
These horned silver-haired ??? were previously unknown

word2vec

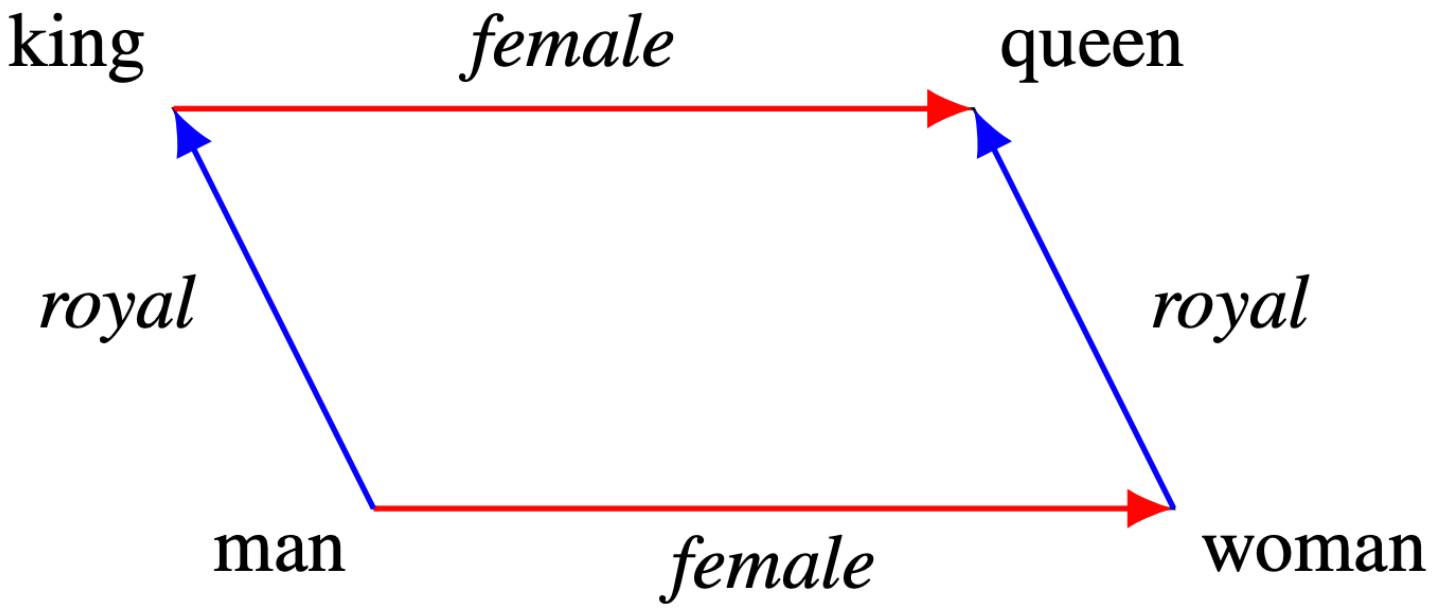
p(*unicorn* | These horned silver-haired ??? were previously unknown)

$$p(x_n \mid x_{-n}) = \text{softmax}(E\, f(x_{-n}))$$



word2vec

- Word embeddings show surprising structure



King – Man + Woman = Queen

Ethayarajh et al. (2018)

word2vec

- Fast and scalable
- Word representations are *independent* of context
  - But interpretation of words depends strongly on context

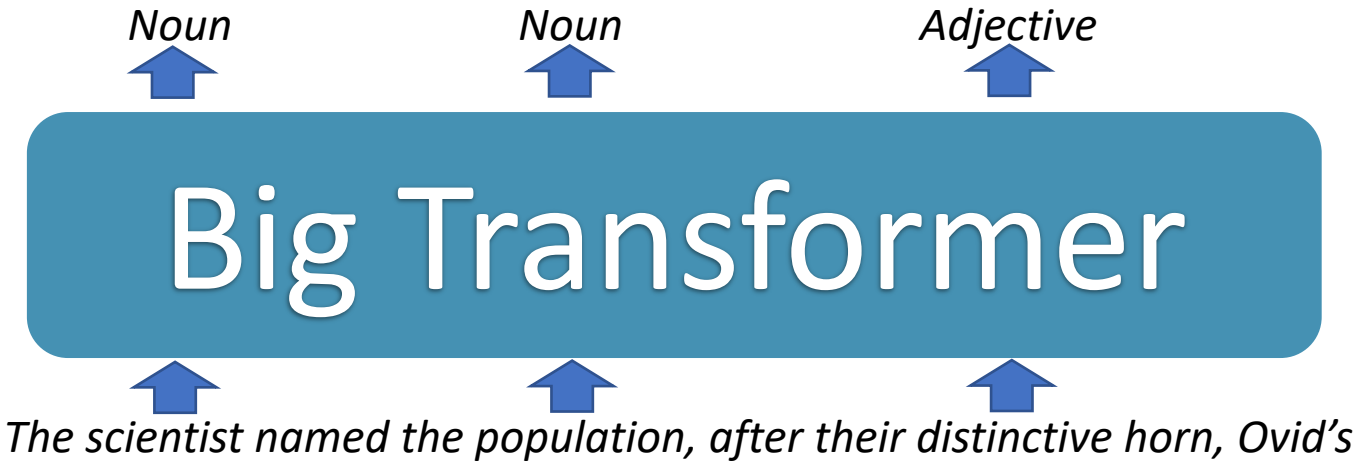
GPT

- Contextualize words by just using left-to-right language model
- Pre-train network to predict the next word
- Fine tune by changing to a task-specific loss function

Radford et al. (2018)

GPT

- Contextualize words by just using left-to-right language model
- Pre-train network to predict the next word
- Fine tune by changing to a task-specific loss function



ELMo

- GPT only considers leftward context
- ELMo pretrains two language models:
  - One left-to-right
  - One right-to-left
  - Concatenate final layer output
- Only ‘shallow’ combination of leftward and rightward context

Peters et al. (2018)

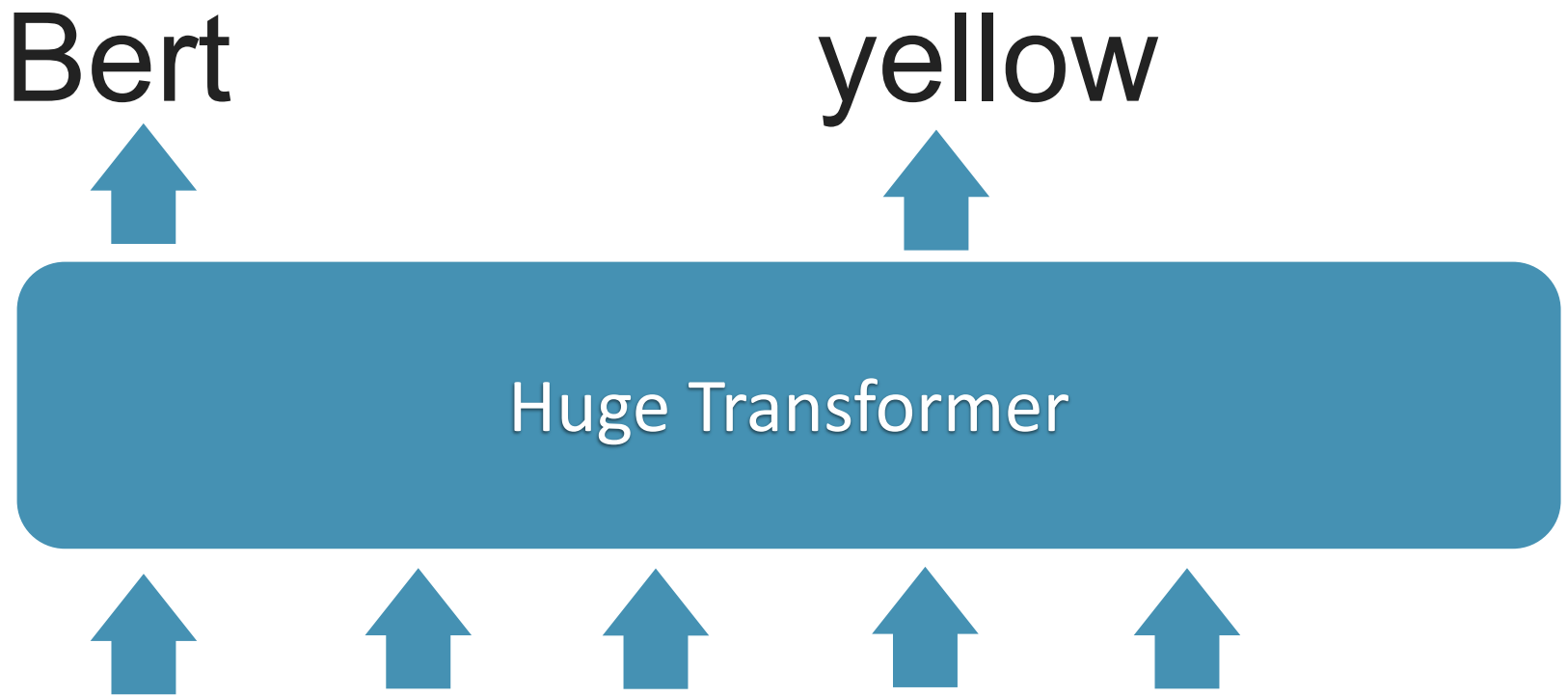
BERT

Huge Transformer

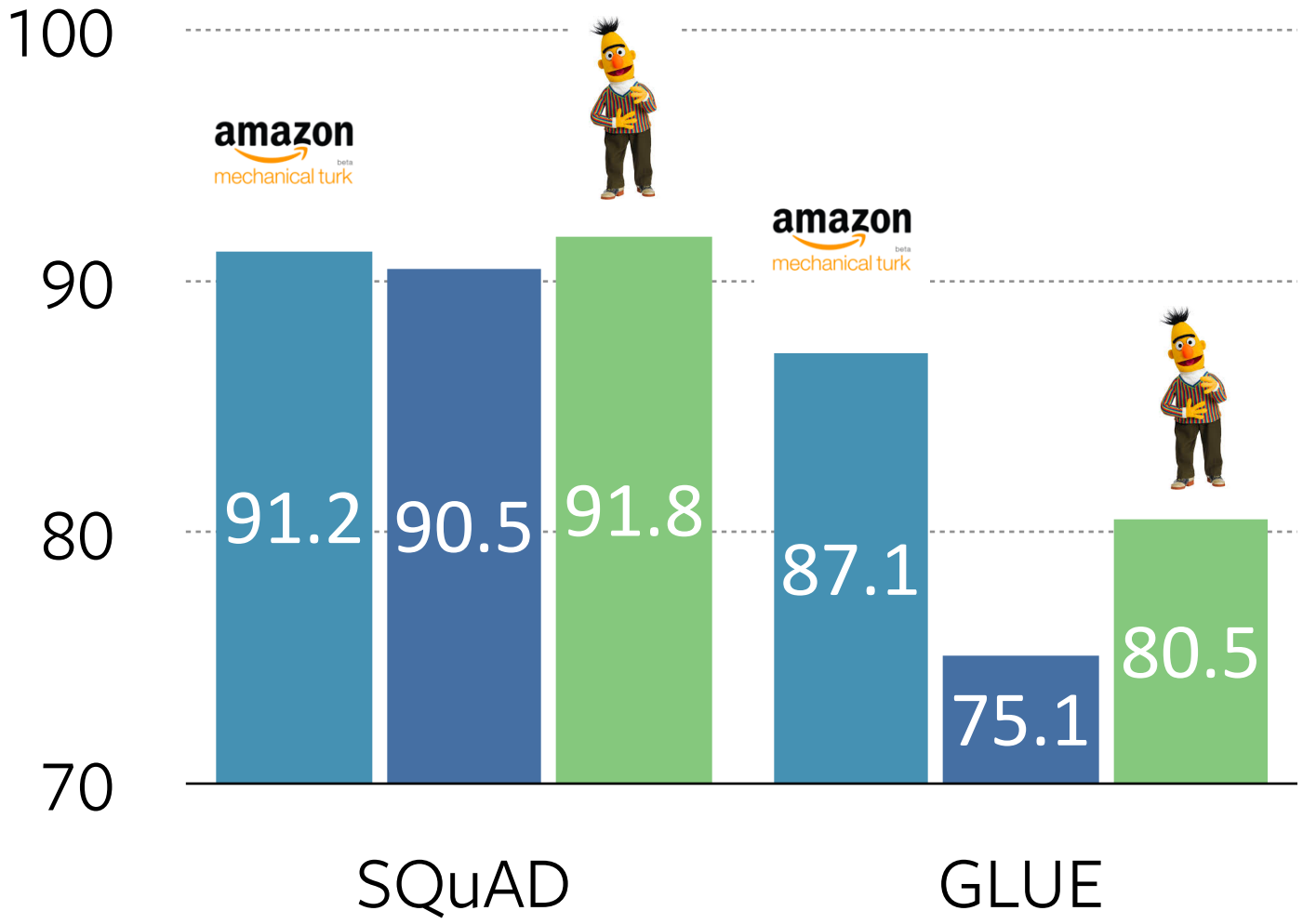
??? is a golden ??? Muppet

Devlin et al. (2018)

BERT



BERT


100
90
80
70
amazon
mechanical turk
91.2
90.5
91.8
amazon
mechanical turk
87.1
75.1
80.5
SQuAD
GLUE

BERT

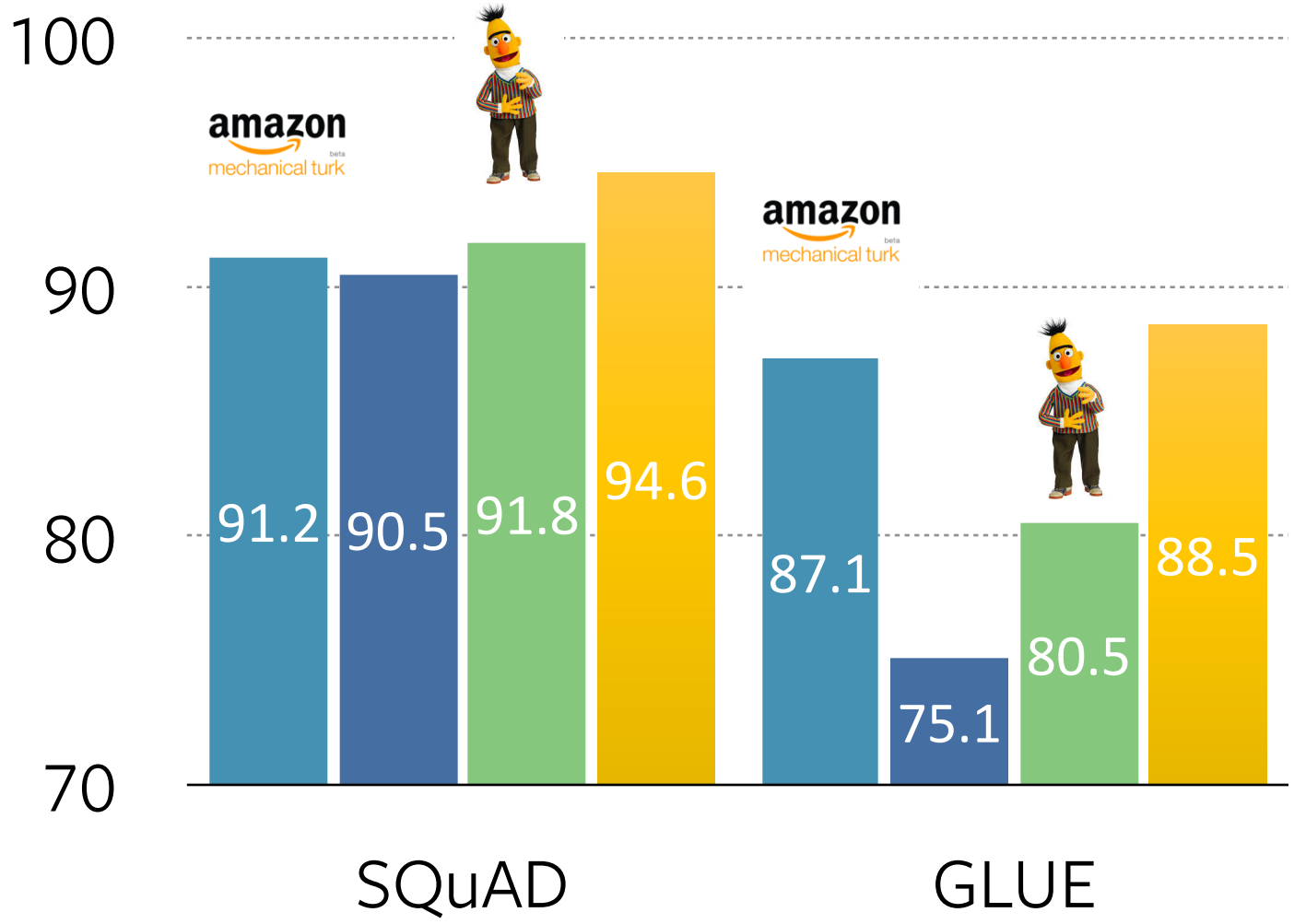
- Mask 15% of tokens, and predict blanks
- All words condition on all other words

RoBERTa

- Simplify BERT's pre-training objective
- Scale up batch sizes
- Train on 1000 GPUs

Liu et al. (2019)

Pre-training for NLP



Pre-training for NLP

- **XLNet:** Predict masked tokens auto-regressively in random order
- **SpanBERT**: Mask spans instead of tokens
- **ELECTRA**: Substitute tokens with similar ones, and predict which changed
- **ALBERT**: Tie weights across layers
- **XLM**: Run BERT on lots of languages

Pre-training for NLP

- Lot of objectives work well!
- Crucial to model deep, bidirectional interactions between words
- Large gains from scaling up pre-training, with no clear limits yet

BART and T5

- How do we pretrain for conditional generation?

BART and T5

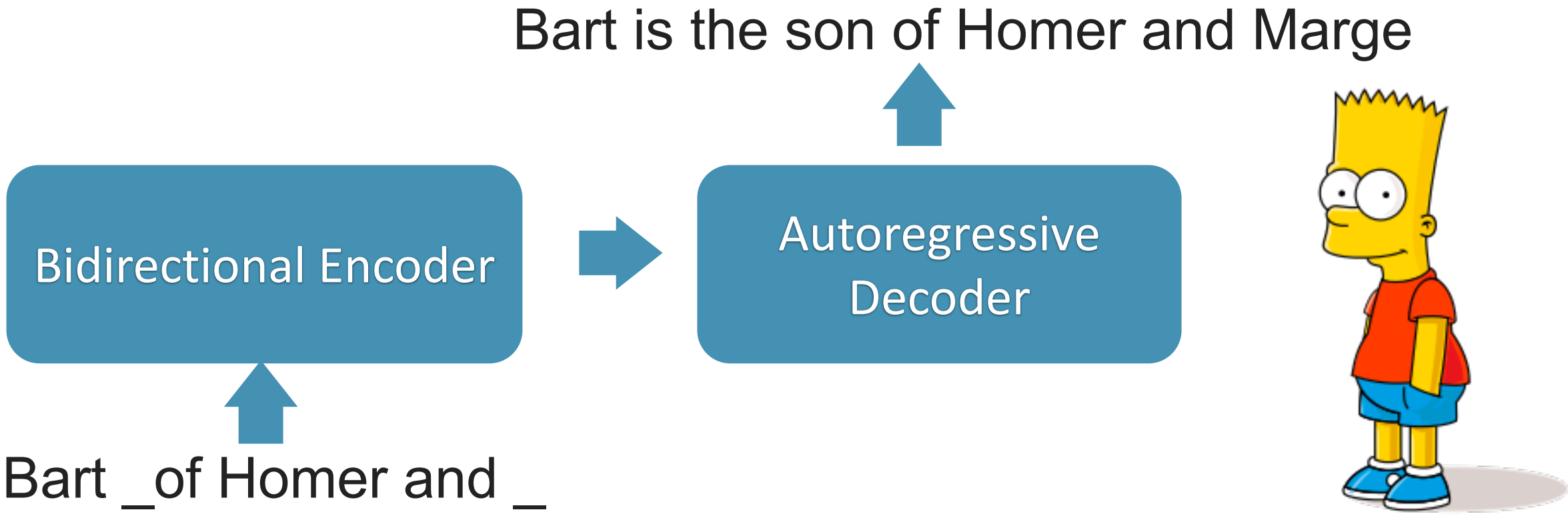
- How do we pretrain for conditional generation?

Pre-training for Conditional Generation

BART: pre-training seq2seq models by de-noising text

Lewis et al. (2019)

Pre-training for Conditional Generation



Lewis et al. (2019)

Pre-training for Conditional Generation

BART: pre-training seq2seq models by de-noising text

- RoBERTa-level performance on SQuAD/GLUE
- New SOTA on summarization, dialogue and abstractive QA datasets

Source Document (abbreviated)	BART Summary
According to Syrian state media, government forces began deploying into previously SDF controlled territory yesterday. ... On October 6, US President Donald Trump and Turkish President Recep Tayyip Erdoan spoke on the phone. Then both nations issued statements speaking of an imminent incursion into northeast Syria ... . On Wednesday, Turkey began a military offensive with airstrikes followed by a ground invasion.	Syrian government forces have entered territory held by the US-backed Syrian Democratic Forces (SDF) in response to Turkey's incursion into the region.

Lewis et al. (2019)

Some open questions

- How should we integrate world knowledge
- How do we model long documents?
  - BERT-based models typically use 512 tokens
- How do we best do multi-task learning?
- Can we fine-tune with less data?
- Are these models really understanding language?

Summary

- Training models on lots of data beats explicitly modelling linguistic structure
  - Architectures should compressing sequences through bottlenecks
- Models can learn a lot about language by predicting words in unlabeled text
  - Fine tuning for specific tasks is then easy
- Bidirectional context is crucial